An algebraic approach to Integer Portfolio problems

F. Castro J. Gago I. Hartillo J. Puerto J.M. Ucha

Abstract

Integer variables allow the treatment of some portfolio optimization problems in a more realistic way and introduce the possibility of adding some natural features to the model.

We propose an algebraic approach to maximize the expected return under a given admissible level of risk measured by the covariance matrix. To reach an optimal portfolio it is an essential ingredient the computation of different test sets (via Gröbner basis) of linear subproblems that are used in a dual search strategy.

1 Introduction

Mean-variance portfolio construction lies at the heart of modern asset management and has been among the most investigated fields in the economic and financial literature. The classical Markowitz's approach, cf. [Mar52], or [Mar00] for a recent reissue of his work, rests on the assumption that investors choose among n risky assets to look for their corresponding weights w_i in their portfolios, on the basis of

1. previously estimated expected returns μ_i, and
2. the corresponding risk of the portfolio measured by the covariance matrix Ω.

Portfolios are considered *mean-variance efficient* in two senses:

- If they minimize the variance for a given admissible return R:

$$\begin{array}{ll} \text{(MVP1)} & \min \begin{pmatrix} w_1 & \dots & w_n \end{pmatrix} \Omega \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}, \\ \text{subject to} & \mu_1 w_1 + \cdots + \mu_n w_n \geq R, \\ & w_1 + \cdots + w_n = 1, \\ & w_j \in \mathbb{R}. \end{array}$$

- If they maximize the expected return for a given admissible risk (variance) r^2:

$$\begin{array}{ll} \text{(MVP2)} & \max \mu_1 w_1 + \cdots + \mu_n w_n, \\ \text{subject to} & \begin{pmatrix} w_1 & \dots & w_n \end{pmatrix} \Omega \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} \leq r^2, \\ & w_1 + \cdots + w_n = 1, \\ & w_i \in \mathbb{R}. \end{array}$$

In problems (MVP1) and (MVP2) the weights w_i stand for the percentage of a given asset in the portfolio. It is well-known that both problems give dual views of a common analysis since they correspond to two scalarizations of the more general bicriteria approach: obtaining the entire set W of mean-variance efficient portfolios. In this sense, (MVP1) and (MVP2) are equivalent in that W can be obtained either solving (MVP1) or (MVP2) parametrically on R (admissible returns) or r (admissible risks), respectively; see [SNT85] for further details.

Although the standard statement of the mean-variance portfolio problem uses continuous variables, there are different reasons to consider integer variables, as it is pointed out in different authors [SM05, LT08, BL07, You98, CF07]. Our goal is to treat problem (MVP2) in its integer version based on the following considerations:

- In real markets, we can buy or sell only an integer amount of assets, so considering real weights in the portfolio is actually an approximation. As it is well known already for linear problems, the rounding of the real values obtained for one optimal portfolio with continuous weights may produce, in principle, an infeasible solution or a very bad approximation to the optimal integer solution. We think that this is indeed the case for a portfolio that potentially considers future contracts, as in [GK08] for commodity futures, to obtain lower correlations concluding on the benefits of diversification. We show this behaviour in Example 5.2.
- The need to diversify the investments in a number of industrial sectors [BL07].
- The constraint of buying stocks by lots [BL07, CF07, JHLM01].
- Another reason for the use of integer variables, usually binary, appears in practice because portfolio managers and their clients often hate small active positions and very large number of assets, the reason being that they produce big transaction and monitoring costs. Hence it is rather usual to add the constraints associated to the following conditions:
  1. there should be at least some previously decided minimum percentage (lower bounds), and
  2. there should be a maximum number of assets (upper bounds).

  Furthermore, transaction costs are included as linear constraints in the return equation, using decision variables, as in [LT08].

The above mentioned requirements enrich any portfolio model for real applications and require 0-1 or, more generally, integer variables. Nevertheless, as far as we know there are no specific algorithms to solve these problems. Note that methods in semi-definite programming deal with this problem, but they are oriented to the continuous case. See also [SM05] for a set of routines that handles these problems in the continuous case. In our approach, it is natural to consider non negative integer variables $x_1, \ldots, x_n$ for the quantities of each product. Then it is necessary to take into account

- the unit prices $a_1, \ldots, a_n$ of the products,
- the expected returns of each stock $\mu_1, \ldots, \mu_n$,
- and the total available budget B.

Then Problem (MVP2), in its integer version, can be restated as

$$\begin{array}{rl} & \max \sum_{i=1}^{n} \mu_i x_i = \boldsymbol{\mu}^t \boldsymbol{x} \\ \text{subject to} & \sum_{i=1}^{n} a_i x_i = \boldsymbol{a}^t \boldsymbol{x} \leq B, \\ & Q(\boldsymbol{x}) \leq B^2 r^2, \\ & \boldsymbol{x} \in (\mathbb{Z}_+)^n, \end{array}$$

where

$$Q(\boldsymbol{x}) = \begin{pmatrix} a_1 x_1 & \dots & a_n x_n \end{pmatrix} \Omega \begin{pmatrix} a_1 x_1 \\ \vdots \\ a_n x_n \end{pmatrix}.$$

1.1 Previous works and our approach

There have been several works with different techniques to treat Problem (MVP1) in an integer framework.

- For (MVP1): piecewise linear approximation in [Sha71] and [Sto73], absolute deviation selection in [YK91], branch-and-cut techniques in [Bie96] and minimax selection approach in [You98].
- For (MVP1) with additional transaction cost: dual Lagrangian relaxation in [MM91, GK87], linear terms transformation in [LC98], separable terms transformation in [Sha71, Sto73], and parallel distributed computation reformulating the objective function into an ellipse and using piecewise linearization in [LT08].

For Problem (MVP2) the literature is not so wide, although this approach is very usual in practice for the so called *benchmark portfolios*, as used in [MM08]. The results of [BPT00] have been in some sense a milestone in applying tools of Algebraic Geometry (namely Gröbner bases) to optimization, although this method is not effective for Problem (MVP2). See [AL94, Ch.1,2], [CLO05, Ch. 2], or [BW05] for introductions to this subject.

Our goal is to present a new algorithm to deal with portfolio problems with integer variables and non-linear constraints. The method is based on the computation of some test sets using Grobner bases. These bases are computed from a linear integer subproblem that contains the original linear constraints together with some new cuts induced by the non-linear constraints. The use of the *reversal test-set* allows us to design a dual search algorithm that moves from the optimal solution of the linear subproblems towards the optimal solution of the entire portfolio problem. Our approach is new with respect to the cited references, and we will see in the last section that it is effective to deal with portfolios with number of stocks comparable to those in the literature (see [CF07, LT08]).

The paper is organized as follows. In Section 2 we fix the formulation of the problem to be treated with the method explained in [TTN95]. In Section 3 the successive additions of linear constraints are explained, and the dual search algorithm based on a test-set computed from a Gröbner basis is applied to find an optimal point of Problem (1). It is also included an illustrative example.

Section 4 explains the existence of a lower bound r_b^2 to the risk value r^2 below that it is not necessary to invest the whole budget to get an optimal portfolio. Section 5 contains some computational experiments and Section 6 draws some conclusions on the paper.

2 Preliminaries

If one accepts the integer version of model (MVP2) to obtain efficient portfolios, the objective function and all constraints but one —that is quadratic— are linear. This is related to the form of the problem treated in [TTN95]. To solve an integer programming problem (**P**) with linear objective function under different linear and nonlinear constraints, the following general approach can be applied (see [TTN95] for a complete example):

1. First obtain a *test-set* for a linear part of (**P**), let us call it (**P′**). A test-set T is a set of vectors in $\mathbb{Z}^n$ such that, given a feasible point F of (**P′**), if none of the feasible points obtained adding the elements of T to F improves the value of the objective function, then F is an optimum of (**P′**). A test-set of a linear integer problem can be obtained via Gröbner bases ([CT91], see [Stu96, ch. 5] for a modern introduction).

   The best known way of obtaining these bases is using programs as `4ti2` ([tt08]), that takes advantage of the special structure of the *toric ideals* corresponding to linear integer problems. Programs for general Gröbner bases are not so good for big examples.

2. Starting at the optimum of the linear problem (**P′**), which is possibly an infeasible point for problem (**P**), use the *reversal* test-set —so decreasing the objective function each time a vector of T is applied— to move throughout the set (tree) of feasible points of (**P′**) until you obtain feasible points *for the whole problem* (**P**). In our case, it means, portfolios with admissible risk. If this happens, one can prune the remaining feasible solutions.

Our approach consists of applying this general idea to Problem (1) mixing it with some bounds obtained from the continuous relaxation of the problem, to reduce the feasible region described by the linear constraints, as in [LT08].

The initial problem is

$$(\mathbf{P}) \quad \max\{\mu^t \boldsymbol{x} \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, Q(\boldsymbol{x}) \leq B^2 r^2, \boldsymbol{x} \in (\mathbb{Z}_+)^n\},$$

and its linear relaxation is

$$(\mathbf{P}') \quad \max\{\mu^t \boldsymbol{x} \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, \boldsymbol{x} \in (\mathbb{Z}_+)^n\}.$$

The purpose of the following section is to explain how to obtain additional linear constraints to improve the accuracy of the linear description of problem (**P**), taking advantage of geometrical properties of the definition of risk. We look for some linear constraints based on the convexity of the hyperquadric given by the covariance matrix Ω, which is symmetric positive definite. However, too many constraints means too many elements in the Gröbner basis, so the point will be to find the precise trade-off between constraints to eliminate unnecessary points in our searching, and at the same time not to increase the basis unnecessarily.

3 A dual search algorithm based on a test-set

A direct approach to problem (**P**) following [TTN95] may lead to a non practical procedure to find the optimum. If we compute a test-set related to problem (**P′**), and move along the set of solutions of the linear relaxation of problem (**P**), the number of points to be processed is huge, even for a small number of variables. The main drawback of the procedure is the great number of integer solutions to be visited between the starting point and the feasible region of the problem (**P**). In order to avoid this enormous enumeration, some cuts can be added, using the convexity of the hyperquadric defined by the symmetric positive definite matrix Q.

We assume that a black box is available providing solutions to linear continuous optimization problems with quadratic convex constraints, as the function `fmincon` in Matlab, the different implementations of semi-definite programming compared in [Mit03], or even the linear time in fixed dimension algorithm by [Dye92].

3.1 Starting tasks

In order to improve our representation of problem (**P**), we proceed as follows:

- The first step is the computation of the continuous solution $\boldsymbol{u}_c$ of the problem

$$\max\{\mu^t \boldsymbol{x} \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, Q(\boldsymbol{x}) \leq B^2 r^2, \boldsymbol{x} \in (\mathbb{R}^+)^n\},$$

  which gives us a return $\boldsymbol{\mu}^t\boldsymbol{u}_c = R_c$. Clearly the discrete return is less than or equal to $\lfloor R_c \rfloor$ (function `ComputeContinuousOptimum` in Algorithm 1).

- Secondly, we need a good discrete feasible point, which will give us a lower bound for the return. The problem (**P**) is always feasible, because the origin belongs to the region, but this point it is not very useful. The rounded point $\boldsymbol{u}_d = \lfloor \boldsymbol{u}_c \rfloor$ is not always feasible, as it is well-known.

- In order to get a feasible starting point, it is possible to decrease each coordinate until we get a feasible point. After that, the point can be improved so that the return cannot be increased in any direction inside the feasible region (function `ComputeDiscreteApprox` in Algorithm 1).

From this point $\boldsymbol{p}_e$ we will reach the discrete optimum. Let $R_e = \boldsymbol{\mu}^t\boldsymbol{p}_e$ be the return associated with the discrete feasible point $\boldsymbol{p}_e$. A new valid formulation of the problem is

$$\max\{\boldsymbol{\mu}^t\boldsymbol{x} \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, Q(\boldsymbol{x}) \leq B^2 r^2, \boldsymbol{x} \in (\mathbb{Z}_+)^n\}.$$

3.2 Addition of new linear constraints

From the above formulation, we improve our description of problem (**P**) in two ways.

1. Adjusting hyperplanes to the hyperquadric given by upper and lower bounds on the variables.

   To this end, for $j = 1, \ldots, n$, we solve the continuous problems (function `ComputeLowerBounds` in Algorithm 1)

$$\min\{x_j \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, Q(\boldsymbol{x}) \leq B^2 r^2, \boldsymbol{x} \in (\mathbb{R}_+)^n\}.$$

   The above minimum values give us an integer lower bound b_j for each variable x_j, applying the ceiling function. The constraints $b_j \leq x_j$ are not going to be involved in the computation of the test-set through the Gröbner basis. This is because we can write $b_j + y_j = x_j$, where $y_j \geq 0$, and this change of variables do not alter the coefficient matrix of the linear cuts, nor the linear cost function. Since the computation of the Gröbner basis does depend only of this matrix, there is no extra computation time.

In a similar way, the maximization problems

$$\max\{x_j \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, Q(\boldsymbol{x}) \leq B^2r^2, \boldsymbol{x} \in (\mathbb{R}_+)^n\}$$

for $j = 1, \ldots, n$, provides us upper bounds. However, these linear constraints highly increase the size of the test-set. We will only use them to fix variables, because the upper and lower bounds of some variables are equal in many examples. This fact allows us reducing the dimensionality of the problem.

We consider the polytope

$$P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, \boldsymbol{\mu}^t\boldsymbol{x} \geq R_e, \boldsymbol{b} \leq \boldsymbol{x}\}$$

where $\boldsymbol{b} \leq \boldsymbol{x}$ stands for the conditions $b_i \leq x_i, i = 1, \ldots, n$.

2. Adding nearly tangent hyperplanes to shrink the polytope.

The main idea is the addition of cuts so that the farthest regions of the polytope P could be cut off. To do that, we use a point of P where the function Q takes its greatest value. This is equivalent to solve the continuous problem

$$\max\{Q(\boldsymbol{x}) \mid \boldsymbol{x} \in P\}.$$

It is coded as function `ComputeMaxRisk` in Procedure NewPolytope. Note that this problem can be efficiently solved since it is of polynomial complexity [KTH79].

Let $\boldsymbol{p}_{\max}$ be a solution of Problem (2), s be the half-line from the feasible point $\boldsymbol{p}_e$ to $\boldsymbol{p}_{\max}$, and $p' = \mathcal{Q} \cap s$ the intersection point of s with the hyperquadric $\mathcal{Q}$ defined by the function $Q(\boldsymbol{x}) = B^2r^2$.

Let H be the supporting hyperplane to $\mathcal{Q}$ at the point p'. By the convexity of $\mathcal{Q}$, the hyperplane H defines a linear half-space that contains the interior of $\mathcal{Q}$.

The coefficients of the hyperplane H are real numbers, so its normal vector $\boldsymbol{n}$ may have non integer components. However, we are looking for linear constraints with integer coefficients, so we round the vector $\boldsymbol{n}$ to an integer vector $\tilde{\boldsymbol{n}} \in \mathbb{Z}^n$ (variable $Prec$ in Procedure NewPolytope). Then we proceed to compute the independent term c of the tangent hyperplane to $\mathcal{Q}$ whose normal vector is equal to $\tilde{\boldsymbol{n}}$, and such that the half-space $\tilde{\boldsymbol{n}}^t\boldsymbol{x} \leq \tilde{c} = \lceil c \rceil$ defines a linear half-space which contains the interior of $\mathcal{Q}$.

This process can be iterated as many times as we wish (Algorithm NewPolytope), shrinking the polytope P. Nevertheless, there should be a trade-off between the number of new hyperplanes and the size of the Gröbner basis associated with the system, so the maximum number of cuts allowed is a parameter of the algorithm. Additionally, the difference between $r_m^2 = \max\{Q(\boldsymbol{x}) \mid \boldsymbol{x} \in P\}$ and the initial risk r_0^2 is another stopping criterion, passed as the parameter Tol to the algorithm.

3.3 Dual iterations with the test-set

Now after the above two phases Problem 1 is transformed to

$$\begin{aligned} & \max \boldsymbol{\mu}^t \boldsymbol{x} \\ \text{s. t.}\quad & \boldsymbol{a}^t \boldsymbol{x} \leq B, \\ & R_e \leq \boldsymbol{\mu}^t \boldsymbol{x} \leq \lfloor R_c \rfloor, \\ & \tilde{\boldsymbol{n}}_k^t \boldsymbol{x} \leq \tilde{c}_k, k = 1, \ldots, s, \\ & \boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \in (\mathbb{Z}_+)^n, \\ & Q(\boldsymbol{x}) \leq B^2 r^2, \end{aligned}$$

where $\tilde{\boldsymbol{n}}_k^t \boldsymbol{x} \leq \tilde{c}_k, k = 1, \ldots, s$ are the new cuts. The test-set G is associated with the linear problem

$$\begin{aligned} & \max \boldsymbol{\mu}^t \boldsymbol{x} \\ \text{s. t.}\quad & \boldsymbol{a}^t \boldsymbol{x} \leq B, \\ & \boldsymbol{\mu}^t \boldsymbol{x} \leq \lfloor R_c \rfloor, \\ & \tilde{\boldsymbol{n}}_k^t \boldsymbol{x} \leq \tilde{c}_k, k = 1, \ldots, s, \\ & \boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \in (\mathbb{Z}_+)^n. \end{aligned}$$

The condition $R_e \leq \boldsymbol{\mu}^t \boldsymbol{x}$ is tested inside the tree-search, and used to prune leaves. The value of R_e is updated as soon as a new feasible point with a better return is found.

Once we have the test-set of the above polytope we proceed with the resolution method. The main bottleneck of our approach is the search over the tree defined by the test-set. The number of points to be processed is strongly related to the initial feasible point $\boldsymbol{p}_e$ found because:

1. The estimated return R_e defines the lowest facet of the polytope P in terms of the objective value.

2. The upper and lower bounds for the variables x_i are strongly determined by R_e. The closer is R_e to the optimal value, the narrower is the interval for each variable x_i.

On the other hand, to apply the reversal test-set search we need an initial (and usually non feasible) point, but not far from feasibility. We take the point $\boldsymbol{p}_{\text{bounds}}$ built by considering the independent terms of the linear constraints of Problem (3.3), after applying the translation $b_i + y_i = x_i$, i.e.,

$$\boldsymbol{p}_{\text{bounds}} = (\boldsymbol{b}, B - \boldsymbol{a}^t \boldsymbol{b}, \lfloor R_c \rfloor - \boldsymbol{\mu}^t \boldsymbol{b}, \tilde{c}_k - \tilde{\boldsymbol{n}}^t \boldsymbol{b})^t.$$

The starting point for the reversal test-set tree search is the point $\boldsymbol{p}_{\text{ini}}$, the reduced of $\boldsymbol{p}_{\text{bounds}}$ by the test-set G. This is the solution of the linear problem (3.3), as shown in [CT91]. With the reversal test-set, we search over the tree of nodes (feasible points for the linear problem) until we obtain a feasible point for the entire problem, including the quadratic constraint (function `TreeSearch` in Algorithm 1). If the switch $SwFictBounds$ is set to true, the search is stopped in the first point that improves the estimated return given by the incumbent point $\boldsymbol{p}_e$.

Although the tree search has to end, a maximum number of processed records is passed to the procedure `TreeSearch` as a parameter. It could happen that the number of points processed exceeds the maximum allowed, and the optimum had not been found. If a new feasible point $\boldsymbol{p}'_e$ is found ($SwImprove =$ true), the bounds can be recomputed. The test-set is still valid for the new search. The only new computations are the independent terms of the hyperplanes and the reduction to find the starting point.

However, if the test-set were huge, it would be better to compute the new linear cuts given by the new estimated point $\boldsymbol{p}'_e$ and its associated Gröbner basis. In general, the Gröbner basis will be shorter, and the elapsed time spent in the tree search will be shortened.

3.4 Restricted search in a region

In the case that a new feasible point is not found after a given number of processed points ($SwImprove =$ false), it is then possible to apply a branch-and-cut technique with the bounds. Indeed, let $\boldsymbol{p}_e$ be the feasible point that gives the value R_e, and $\boldsymbol{b}$ the vector of lower bounds for the variables $x_i, i = 1, \dots, n$. Compute a point $\boldsymbol{b}' = \boldsymbol{b} + \alpha(\boldsymbol{p}_e - \boldsymbol{b}), 0 < \alpha < 1$ (usually $\alpha = 1/2$), which we call fictitious bound, and consider the following problem:

$$\begin{array}{rl} & \max \boldsymbol{\mu}^t\boldsymbol{x} \\ \text{s. t.} & \boldsymbol{a}^t\boldsymbol{x} \leq B, \\ & R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, \\ & \tilde{\boldsymbol{n}}_k^t\boldsymbol{x} \leq \tilde{c}_k, k = 1, \dots, s, \\ & \boldsymbol{x} \geq \boldsymbol{b}', \boldsymbol{x} \in (\mathbb{Z}_+)^n, \\ & Q(\boldsymbol{x}) \leq B^2 r^2. \end{array} \tag{1}$$

Solving the above problem, we expect to find a new feasible point to relaunch the search process. The idea is simple: the search is restricted to a smaller region, but the solution of the original problem is not guaranteed to be in that region. It is a heuristic technique to take advantage that this new problem does not need a new test-set. In our implementation, this search can be launched by setting the switch $SwFictBounds$ equal to true. The process is stopped as soon as a new point is found, and then we start again. If no point is found after the maximum number of allowed nodes (variable $MaxNumNodes$ in Algorithm 1), then we stop, and the point $\boldsymbol{p}_e$ is our best value.

The pseudocode of the main algorithm is described in `DiscreteOptimum`. The procedure `NewPolytope` presents the pseudocode of the strengthening of the polytope P by adding valid cuts. The switch $SwEOP$ is used to mark the end of the process.

3.5 An illustrative example

Let $\boldsymbol{a} = (6075, 3105)^t$ be the vector of prices, and $\boldsymbol{\mu} = (12500, 10000)^t$ the vector of returns, with the covariance matrix equals to

$$\Omega = \begin{pmatrix} .832843e-4 & .485325e-4 \\ .485325e-4 & .651298e-3 \end{pmatrix}.$$

Let $B = 9 \times 10^6$ be the budget, and $r^2 = 3 \times 10^{-5}$ the fixed risk. The continuous optimum is $\boldsymbol{u}_c = (772.754778, 215.028056)^t$, with a total return $R_c = 11809715.29$. Then $\lfloor R_c \rfloor = 11809715$, and rounding $\boldsymbol{u}_c$ we get the point $\boldsymbol{u}_d = (773, 215)$, which is not a feasible point. Subtracting from the components, we eventually reach a feasible point $\boldsymbol{p}_e = (773, 214)$, whose associated return is $R_e = 11802500$. The lower bounds b_1 and b_2 are now computed, solving first the continuous problems

$$\min \{x_i \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq R_c, Q(\boldsymbol{x}) \leq r^2B^2, \boldsymbol{x} \in (\mathbb{R}^+)^2\},$$

where

$$Q(\boldsymbol{x}) = \begin{pmatrix} a_1x_1 & a_2x_2 \end{pmatrix} \Omega \begin{pmatrix} a_1x_1 \\ a_2x_2 \end{pmatrix}. \tag{2}$$

The respective continuous values are $\tilde{b}_1 = 752.69$, rounded to $b_1 = 753$, and $\tilde{b}_2 = 190.58$, rounded to $b_2 = 191$. We want to solve the problem

$$\max \{\boldsymbol{\mu}^t\boldsymbol{x} \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, Q(\boldsymbol{x}) \leq r^2B^2, \boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \in (\mathbb{Z}_+)^2\}.$$

In the associated linear problem

$$\max\{\boldsymbol{\mu}^t\boldsymbol{x} \mid \boldsymbol{\mu}^t\boldsymbol{x} + z_1 = \lfloor R_c \rfloor, \boldsymbol{a}^t\boldsymbol{x} + z_2 = B, \boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \in (\mathbb{Z}_+)^2\}$$

we change the variables $x_i = y_i + b_i$, and the resulting linear problem has the same coefficient matrix. The computation of a Gröbner basis leads to the test-set formed by vectors

$$\begin{array}{lll} \boldsymbol{v}_1 = (-4, 5, 8775, 0)^t, & \boldsymbol{v}_2 = (-1, 1, 2970, 2500)^t, & \boldsymbol{v}_3 = (0, -1, 3105, 10000)^t, \\ \boldsymbol{v}_4 = (1, -2, 135, 7500)^t, & \boldsymbol{v}_5 = (2, -3, -2835, 5000)^t, & \boldsymbol{v}_6 = (3, -4, -5805, 2500)^t. \end{array}$$

Now, we reduce the point

$$\boldsymbol{p}_{\text{bounds}} = (b_1, b_2, B - a_1b_1 - a_2b_2, \lfloor R_c \rfloor - \mu_1b_1 - \mu_2b_2)^t = (753, 191, 3832470, 487215)^t$$

with the test-set to get the linear problem optimum, which is the starting point $\boldsymbol{p}_{\text{ini}}$ of the tree search. In this case, $\boldsymbol{p}_{\text{ini}} = (791, 192, 3598515, 2215)$. We now show the path followed by the search procedure in the tree of solutions of the linear part of the problem. In each node, we write the distance Δ_1 to the continuous return associated with it, that is, $\Delta_1 = \lfloor R_c \rfloor - \boldsymbol{\mu}^t(\boldsymbol{p} + \boldsymbol{v}_i)$. The initial distance is $\Delta_e = 7215$, the difference between $\lfloor R_c \rfloor$ and R_e. The larger the value, the smaller the return. Therefore, values larger than Δ_e means that the corresponding branch can be pruned. The list of nodes to be processed are then ordered by Δ_1. Note that black dots '$\bullet$' mean pruned nodes, and white dots '$\circ$' mean new nodes. The points are shortened to the two first components to save space.

- $\star$ Node $\boldsymbol{p} = (791, 192)^t, \Delta_e = 7215$. Leaves $\boldsymbol{p} + \boldsymbol{v}_i, i = 1, \ldots, 6$:
  - $\boldsymbol{p} + \boldsymbol{v}_1 = (787, 197)^t, \Delta_1 = 2215, r^2 \geq r_0^2$. New node $\circ$.
  - $\boldsymbol{p} + \boldsymbol{v}_2 = (790, 193)^t, \Delta_1 = 4715, r^2 \geq r_0^2$. New node $\circ$.
  - $\boldsymbol{p} + \boldsymbol{v}_3 = (791, 191)^t, \Delta_1 = 12215 > \Delta_e$. Pruned by Δ_e.
  - $\boldsymbol{p} + \boldsymbol{v}_4 = (792, \underline{190})^t, \Delta_1 = 9715$. Pruned by bound $b_2 = 191$.
  - $\boldsymbol{p} + \boldsymbol{v}_5 = (793, \underline{189})^t, \Delta_1 = 7215$. Pruned by bound $b_2 = 191$.
  - $\boldsymbol{p} + \boldsymbol{v}_6 = (794, \underline{188})^t, \Delta_1 = 4715$. Pruned by bound $b_2 = 191$.

  The above information gives rise to the following descendants.

$$\begin{tikzcd}
\circ(791,192) \arrow[r] \arrow[d] \arrow[dr] \arrow[drr] \arrow[drrr] \arrow[drrrr] & \bullet(794,\underline{188}) & & & \\
\circ(787,197) & \circ(790,193) & \bullet(791,191) & \bullet(792,\underline{190}) & \bullet(793,\underline{189})
\end{tikzcd}$$

  List of ordered nodes: $\{(787, 197)^t, (790, 193)^t\}$.

- $\star$ Node $\boldsymbol{p} = (787, 197)^t, \Delta_e = 7215$. Leaves $\boldsymbol{p} + \boldsymbol{v}_i, i = 1, \ldots, 6$:
  - $\boldsymbol{p} + \boldsymbol{v}_1 = (783, 202)^t, \Delta_1 = 2215, r^2 \geq r_0^2$. New node $\circ$.
  - $\boldsymbol{p} + \boldsymbol{v}_2 = (786, 198)^t, \Delta_1 = 4715, r^2 \geq r_0^2$. New node $\circ$.
  - $\boldsymbol{p} + \boldsymbol{v}_3 = (787, 196)^t, \Delta_1 = 12215 > \Delta_e$. Pruned by Δ_e.
  - $\boldsymbol{p} + \boldsymbol{v}_4 = (788, 195)^t, \Delta_1 = 9715 > \Delta_e$. Pruned by Δ_e.

- $\boldsymbol{p}+\boldsymbol{v}_5 = (789,194)^t, \Delta_1 = 7215 = \Delta_e$. Pruned by Δ_e.
- $\boldsymbol{p}+\boldsymbol{v}_6 = (790,193)^t$. Already in the list of nodes to be processed.

The corresponding diagram is displayed as

$$\begin{tikzcd}
\circ(787,197) \arrow[r] \arrow[d] \arrow[dr] \arrow[drr] \arrow[drrr] \arrow[drrrr] & \bullet(790,193) & & & \\
\circ(783,202) & \circ(786,198) & \bullet(787,196) & \bullet(788,195) & \bullet(789,194)
\end{tikzcd}$$

List of ordered nodes $\{(783,202)^t, (786,198)^t, (790,193)^t\}$.

⋆ Node $\boldsymbol{p} = (783,202)^t, \Delta_e = 7215$. Leaves $\boldsymbol{p}+\boldsymbol{v}_i, i = 1,\dots,6$:

- $\boldsymbol{p}+\boldsymbol{v}_1 = \boxed{(779,207)^t}$, $\Delta_1 = 2215 < \Delta_e, r^2 \leq r_0^2$. Feasible point, and improvement. Update $\Delta_e = 2215$.
- $\boldsymbol{p}+\boldsymbol{v}_2 = (782,203)^t, \Delta_1 = 4715 > \Delta_e$. Pruned by Δ_e.
- $\boldsymbol{p}+\boldsymbol{v}_3 = (783,201)^t, \Delta_1 = 12215 > \Delta_e$. Pruned by Δ_e.
- $\boldsymbol{p}+\boldsymbol{v}_4 = (784,200)^t, \Delta_1 = 9715 > \Delta_e$. Pruned by Δ_e.
- $\boldsymbol{p}+\boldsymbol{v}_5 = (785,199)^t, \Delta_1 = 7215 > \Delta_e$. Pruned by Δ_e.
- $\boldsymbol{p}+\boldsymbol{v}_6 = (786,198)^t, \Delta_1 = 4715 > \Delta_e$. Deleted of the list of nodes to be processed.

The tree representation is

$$\begin{tikzcd}
\circ(783,202) \arrow[r] \arrow[d] \arrow[dr] \arrow[drr] \arrow[drrr] \arrow[drrrr] & \bullet(786,198) & & & \\
\boxed{(779,207)} & \bullet(782,203) & \bullet(783,201) & \bullet(784,200) & \bullet(785,199)
\end{tikzcd}$$

List of ordered nodes $\{(790,193)^t\}$.

⋆ Node $\boldsymbol{p} = (790,193)^t$. This node is pruned because $\Delta_1 > \Delta_e$, so the process is finished. The total number of processed nodes is 4.

The optimum is $(779,207)^t$, with a difference return of 2215 units from the continuous solution. The initial estimated point was at 7215 units. This example shows a large difference between the rounded solution and the discrete optimum.

Although this example is very simple, we will now show the improvement that we get using the cuts. The initial polytope P is defined by

$$P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, \boldsymbol{x} \geq \boldsymbol{b}\}.$$

The first maximum of the quadric $Q(\boldsymbol{x})$ described in Equation (2) on the polytope is $\boldsymbol{p}_{\max} = (753, \frac{479443}{2000})^t$ and the tangent line at the intersection point has as normal vector $(0.54452, 0.45547)^t$. Rounding to three digits, the new normal vector of the hyperplane is $(545,455)^t$, and the independent term is 519113.7265. Then the first cut is $545x_1 + 455x_2 \leq 519114$. Adding this inequality to the polytope P, and repeating the process, we have the new maximum at point

$\boldsymbol{p}_{\max} = (\frac{1979943}{2500}, 191)^t$, and the normal vector of the tangent cut is equal to $(0.56698, 0.43301)^t$. Then the new cut is $567x_1 + 433x_2 \leq 531402$.

The linear problem, with the slack variables considered, is

$$\begin{array}{rl} & \max \boldsymbol{\mu}^t \boldsymbol{x} \\ \text{s. t.} & \boldsymbol{\mu}^t\boldsymbol{x} + z_1 = \lfloor R_c \rfloor, \\ & \boldsymbol{a}^t\boldsymbol{x} + z_2 = B, \\ & \boldsymbol{x} \geq \boldsymbol{b}, \\ & 545x_1 + 455x_2 + z_3 = 519114, \\ & 567x_1 + 433x_2 + z_4 = 531402, \end{array}$$

and the test-set associated with it has 7 elements:

$$\begin{array}{ll} \boldsymbol{w}_1 = (-4, 5, 8775, 0, -95, 103)^t, & \boldsymbol{w}_2 = (-1, 1, 2970, 2500, 90, 134)^t, \\ \boldsymbol{w}_3 = (0, -1, 3105, 10000, 455, 433)^t, & \boldsymbol{w}_4 = (1, -2, 135, 7500, 365, 299)^t, \\ \boldsymbol{w}_5 = (2, -3, -2835, 5000, 275, 165)^t, & \boldsymbol{w}_6 = (3, -4, -5805, 2500, 185, 31)^t, \\ \boldsymbol{w}_7 = (7, -9, -14580, 2500, 280, -72)^t. & \end{array}$$

The starting point $\boldsymbol{p}_{\text{ini}}$ is the reduction of the point

$$\boldsymbol{p}_{\text{bounds}} = (753, 191, 3832470, 487215, 21824, 21748)^t$$

with the test-set. In this case, $\boldsymbol{p}_{\text{ini}} = (\mathbf{779}, \mathbf{207}, 3624840, 2215, 374, 78)^t$, and it is already the optimum point.

In this example, there is a large amount of money (value 3624840) that is not invested. The explanation of this counterintuitive behaviour is because the risk r^2 is below a critical threshold r_b^2. The way to compute this threshold r_b^2 is the goal of the next section.

4 A remark on admisible risks

In Problem (MVP2), there exist values of the risk r^2 where the optimal investment does not exhaust the available budget, i.e., the linear constraint $\boldsymbol{a}^t\boldsymbol{x} = B$ is not active in the optimal solution. Furthermore, there exists a value r_b^2 (*border risk*) below that it is not necessary to invest the whole budget to get the optimum.

The main idea is that the optimum of a linear function with a quadratic constraint $Q(\boldsymbol{x}) \leq B^2r^2$, Q symmetric positive definite matrix, is found at the point on the quadric whose tangent hyperplane is parallel to the vector given by the objective function. The only problem is dealing with the negative components that the point could have.

Proposition 4.1. *In Problem (MVP2), there exists a risk r_b^2 such that if $r^2 < r_b^2$, then the optimal investment does not need to invest the overall budget.*

Proof. Given a quadric $\mathcal{C}$ with matrix

$$\mathcal{Q} = \begin{pmatrix} a_{00} & \boldsymbol{a}_0^t \\ \boldsymbol{a}_0 & Q_{00} \end{pmatrix}, Q_{00} \text{ symmetric positive definite matrix,}$$

and $\boldsymbol{v}$ the normal vector of the hyperplane $\boldsymbol{v}^t\boldsymbol{x} = 0$, we are looking for a point $p \in \mathcal{C}$ such that the hyperplane

$$\begin{pmatrix} 1 & p^t \end{pmatrix} \mathcal{Q} \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix} = 0$$

is parallel to $\boldsymbol{v}^t\boldsymbol{x} = 0$. Then $\boldsymbol{a}_0 + Q_{00}p = \lambda\boldsymbol{v}$ for certain $\lambda \in \mathbb{R}$, and applying that p belongs to $\mathcal{C}$ we get

$$\lambda^2 = \frac{\boldsymbol{a}_0^t Q_{00}^{-1}\boldsymbol{a}_0 - a_{00}}{\boldsymbol{v}^t Q_{00}^{-1}\boldsymbol{v}}, p = Q_{00}^{-1}(\lambda\boldsymbol{v} - \boldsymbol{a}_0).$$

There are two solutions in λ, and we hold the positive one. In the case of Problem (MVP2), we have $\boldsymbol{a_0} = \boldsymbol{0}$, $a_{00} = -r^2B^2$, and $Q_{00} = C = D\Omega D$, where D is the diagonal matrix $\mathrm{diag}(a_1, \ldots, a_n)$. The quadric $\mathcal{C}$ is centered at the origin, the vector $\boldsymbol{v}$ is now $\boldsymbol{\mu}$, and

$$\lambda = \frac{rB}{\sqrt{\boldsymbol{\mu}^t C^{-1}\boldsymbol{\mu}}} \text{ and the tangent point is } p_t = \frac{rB}{\boldsymbol{\mu}^t C^{-1}\boldsymbol{\mu}} C^{-1}\boldsymbol{\mu}.$$

This point could have negative components, which means that it is outside of the feasible region of the problem. Let J be the set of indexes j such that the j-th component of vector $C^{-1}\boldsymbol{\mu}$ is positive. Let C_J be the hyperquadric restricted to the intersection of hyperplanes $x_j = 0, j \in J$, and $\boldsymbol{\mu}_J$ the vector of components $\boldsymbol{\mu}_j$ with $j \in J$. The restricted hyperquadric is centered at the origin, an the optimum of the restricted problem is reached at

$$q_t = \alpha C_J^{-1}\boldsymbol{\mu}_J, \text{ where } \alpha = \frac{rB}{\sqrt{\boldsymbol{\mu}_J^t C_J^{-1}\boldsymbol{\mu}_J}}.$$

The total amount of invested money is the dot product $q_t \bullet \boldsymbol{a}_J$, and it must be less than B:

$$q_t \bullet \boldsymbol{a}_J = \frac{rB}{\sqrt{\boldsymbol{\mu}_J^t C_J^{-1}\boldsymbol{\mu}_J}} \boldsymbol{a}_J^t C_J^{-1}\boldsymbol{\mu}_J < B.$$

Then

$$r^2 < \frac{\boldsymbol{\mu}_J^t C_J^{-1}\boldsymbol{\mu}_J}{(\boldsymbol{a}_J^t C_J^{-1}\boldsymbol{\mu}_J)^2} = r_b^2.$$

□

It is worth noting that r_b^2 does not depend on the initial budget B.

5 Computational results

This section illustrates the use of our approach in solving some integer portfolio problems with data taken from the literature. In doing that, we have implemented Algorithm 1 in Scilab, to get a portable code, in an Intel Core2 Duo CPU, 2.53 GHz, and 3 GB of RAM (code is available upon request for comparison purposes). The first example solves an actual problem with 44 stocks, whereas the second one shows the big sensitivity of these models with regard to the use of rounded solutions from the optimal solutions of the relaxed (continuous) formulation.

Example 5.1. This example illustrates the use of our methodology with actual data taken from 44 stocks indexed in Eurostoxx, from January 2003 to December 2007. The vector of initial prices is given by the prices of the stocks on January 3rd 2008 (see Table 1), and the returns are estimated from the monthly historical data.

Ticker	Price	Return	Ticker	Price	Return	Ticker	Price	Return
aca.pa	22.7	2.8	agn.as	12.0	0.8	ai.pa	101.6	12.4
alv.de	144.9	32.6	bas.de	100.9	24.9	bay.de	61.6	23.0
bbva.mc	16.6	2.9	bibe.mc	10.1	2.4	bn.pa	61.2	10.9
bnp.pa	73.1	11.0	ca.pa	52.2	5.4	cs.pa	27.0	5.6
dai.de	64.7	13.5	db1.de	128.6	45.8	dbk.de	87.8	17.8
dg.pa	48.7	14.1	dte.de	14.9	1.3	enel.mi	8.1	0.7
eni.mi	25.1	3.4	eoan.de	143.8	37.8	fora.as	18.2	1.9
fp.pa	56.6	7.8	fte.pa	24.5	1.5	g.mi	30.6	2.2
gle.pa	97.6	15.8	gsz.pa	39.5	7.9	ing.as	26.1	5.1
isp.mi	5.3	1.3	lvmh.pa	82.0	13.9	muv2.de	132.0	19.0
noa3.de	25.4	4.7	or.pa	96.2	9.4	phia.as	28.6	4.3
rep.mc	24.9	3.7	rno.pa	95.2	20.0	rwe.de	95.0	28.0
san.mc	14.6	3.1	san.pa	62.0	4.7	sap.de	34.5	4.5
sgo.pa	62.4	13.0	sie.de	107.1	24.8	su.pa	91.1	17.9
tef.mc	21.9	4.5	ucg.mi	5.6	0.7			

Table 1: Data from EuroStoxx

r_0^2	$r_{\max}^2$	cuts	basis	reduction	nodes	optimum
0.0015	0.00154	1	6657		165	$x_6 = 42, x_{14} = 4, x_{16} = 29,$
			(9 s.)	(22 s.)	(99 s.)	$x_{20} = 5, x_{28} = 1, x_{36} = 8.$
0.0020	0.00205	1	40256		137	$x_6 = 51, x_{14} = 5, x_{16} = 19,$
			(446 s.)	(240 s.)	(497 s.)	$x_{20} = 1, x_{28} = 1, x_{36} = 12.$
0.0025	0.00256	2	27082		32	$x_6 = 59, x_{14} = 6, x_{16} = 13,$
			(194 s.)	(421 s.)	(83 s.)	$x_{28} = 2, x_{36} = 10.$
0.0030	0.00301	1	12504		62	$x_6 = 64, x_8 = 1, x_{14} = 8,$
			(26 s.)	(142 s.)	(81 s.)	$x_{16} = 1, x_{28} = 1, x_{36} = 10, x_{37} = 1$
0.0035	0.00351	1	2357		0	$x_6 = 68, x_{14} = 10, x_{16} = 1, x_{36} = 5.$
			(1 s.)	(4 s.)	(0 s.)	
0.0040	0.00430	0	569		9844	$x_6 = 74, x_{14} = 10, x_{16} = 1,$
			(0 s.)	(6 s.)	(821 s.)	$x_{28} = 2, x_{36} = 1.$
	0.00404	1	11924		11	$x_6 = 74, x_{14} = 10, x_{16} = 1,$
			(32 s.)	(121 s.)	(10 s.)	$x_{28} = 2, x_{36} = 1.$
0.0045	0.00451	1	7087		0	$x_6 = 84, x_{14} = 6, x_{16} = 1,$
			(14 s.)	(33 s.)	(0 s.)	$x_{28} = 1.$
0.0050	0.00508	0	357		6	$x_6 = 91, x_{14} = 3, x_{28} = 1.$
			(1 s.)	(0 s.)	(0 s.)	

Table 2: Discrete optimums for different values of risk r_0^2

The border risk is $r_b^2 = 0.00035$, and $B = 6000$. For the computations, all the prices and returns have been multiplied by 10 in order to work with integer variables. In this example we set the parameters to

$$r_{\max}^2 - r_0^2 < 10^{-4}, MaxNumCuts = 4, MaxNumNodes = 10000$$

In Table 2 we show the results of the application of our algorithm assuming different risk levels. Each risk level is organized in a block of two rows. The first one gives the corresponding information, and below we report on the elapsed time to obtain these elements. Column $r_{\max}^2$ contains the greatest risk reached in the polytope with the number of added tangents as shown by the following column. Column 'basis' denotes the number of elements of the computed test-set, and column 'processed' contains the number of new nodes found and explored. Finally, column 'optimum' is a feasible point with the best return. The time in seconds of these tasks appears in parenthesis under the columns 'basis', 'reduction' and 'nodes', respectively.

It is worth remarking the case $r^2 = 0.0040$. The first row shows the number of processed nodes until an optimal point was reached, with no added cutting hyperplane. The second row gives us an example of the effectiveness of adding new cuts. The test-set is computed very quickly, although the number of elements is big. However, the number of processed points is very small, and hence it is also small the total elapsed time.

$r_{\max}^2$	tang.	basis	reduction	$Sw1$	$Sw2$	nodes	improvement
0.00118	3	32495		0	1	max	
		(244 s.)	(333 s.)			(426 s.)	
				1	0	88	$x_6 = 35, x_8 = 36, x_{14} = 2,$
			(176 s.)			(0 s.)	$x_{16} = 27, x_{20} = 8, x_{28} = 31,$
							$x_{35} = 3, x_{36} = 3, x_{43} = 1$
0.00107	4	16930		0	1	max	$x_6 = 34, x_8 = 22, x_{14} = 2,$
		(52 s.)	(130 s.)			(393 s.)	$x_{16} = 29, x_{20} = 9, x_{28} = 24,$
							$x_{35} = 3, x_{36} = 3, x_{43} = 1$
							reached at 2894 nodes in 48 s.
0.00107	4	18637		0	1	max	
		(49 s.)	(114 s.)			(439 s.)	
				1	0	9759	$x_6 = 33, x_8 = 29, x_{14} = 2,$
			(80 s.)			(785 s.)	$x_{16} = 31, x_{20} = 9, x_{28} = 26,$
							$x_{35} = 2, x_{36} = 3$
0.00105	4	14670		0	1	max	$x_6 = 34, x_8 = 32, x_{14} = 2,$
		(28 s.)	(79 s.)			(627 s.)	$x_{16} = 29, x_{20} = 8, x_{28} = 10,$
							$x_{35} = 3, x_{36} = 4, x_{43} = 2$
							reached at 1680 nodes in 29 s.
0.00101	2	2613		0	0	2648	$x_6 = 34, x_8 = 32, x_{14} = 2,$
		(1 s.)	(10 s.)			(853 s.)	$x_{16} = 29, x_{20} = 8, x_{28} = 10,$
							$x_{35} = 3, x_{36} = 4, x_{43} = 2$
							optimum

Table 3: Steps in the computation for $r_0^2 = 0.0010$

Table 3 contains all the iterations done in the computation of the optimum for the case $r_0^2 = 0.0010$. The column $Sw1$ refers to the variable $SwFictBounds$, and $Sw2$ to $SwNumNodes$.

The first one is true when fictitious bounds are used, and the second one is true when the number of processed records is greater than $MaxNumNodes = 10000$. The column 'improvement' contains a new point found with better return than the initial point. Again, the time in seconds of each task appears in parenthesis.

Example 5.2. This example is devoted to show the difference between the rounded continuous solution and the integer solution of a portfolio problem. It consists of a mixing of usual stocks (Microsoft and General Electric) with the value of a future contract based on oil, as in [Chn09], so the number of values is $n = 3$. The initial data is given by

Stock	Price (a_i)	Return (μ_i)
MSFT	35.22	3.64
GE	36.76	3.64
Oil	4000	10000

and the covariance matrix is

$$\Omega = \begin{pmatrix} 0.003250634 & 0.000654331 & 0.022513263 \\ 0.000654331 & 0.001578359 & -0.006610861 \\ 0.022513263 & -0.006610861 & 26.35846804 \end{pmatrix}.$$

We fix the risk to $r_0^2 = 1.52$, and compute the optimum for different budgets B. The test-set associated with the constraints

$$\boldsymbol{a}^t\boldsymbol{x} \leq B, R_e \leq \boldsymbol{\mu}^t\boldsymbol{x} \leq \lfloor R_c \rfloor, \boldsymbol{x} \in (\mathbb{Z}_+)^n$$

has 2663 elements. The basis remains equal for all the considered cases. However, the capacity of computation is run out when only one more cut is added. If we take as initial point $\boldsymbol{p}_e$, the discrete approximation given by rounding, the tree searching was unable to reach the optimal point for $MaxNumNodes = 50000$. This fact is reported as 'E' in Table 4

The function `ComputeDiscreteApprox` should be changed to get a better discrete approximation than the rounded value. If we take an approximation based on the best value for the future contract, we get Table 4.

It is easy to see the enormous difference between the return of the discrete approximation and the corresponding return of the discrete optimum for each budget.

6 Conclusions

We have presented an algorithm to deal with portfolio problems with integer variables and non-linear constraints. The presented model was not previously treated as far as we know. The method is based on the computation of some test sets using Gröbner bases, an algebraic tool. These Gröbner bases are computed from a linear integer subproblem that contains the original linear constraints and some new cuts induced by the non linear constraints. The reversal test-set, given by the Gröbner basis, allows us to perform a dual search algorithm from the optimal solution of the linear subproblem towards the optimal solution of the whole portfolio problem. This technique has allowed us to solve problems of size similar to the exposed in [CF07, LT08].

Budget	continuous optimum	return	nodes	optimum	R_d
50000	$(1079.87, 0, 2.99)$	33848.34			
	discrete approx.				
	$(1192, 0, 2)$	$\boxed{24338.88}$	E		
	$(219, 824, 3)$	33796.52	6066	$(314, 705, 3)$	**33815.06**
75000	$(1619.80, 0, 4.49)$	50772.52			
	discrete approx.				
	$(1675, 0, 4)$	46097.00	22790	$(1675, 0, 4)$	46097.00
100000	$(2159.74, 0, 5.98)$	67696.69			
	discrete approx.				
	$(2271, 0, 5)$	$\boxed{58266.44}$	E		
	$(439, 1646, 6)$	67596.69	22991	$(687, 1409, 6)$	**67629.44**

Table 4: Mixed example

Acknowledgments

This research was supported by grants P06-FQM-01366 and FQM-333 (Junta de Andalucía), MTM2007-64509 and MTM2007-67433-C02-01 (Ministerio de Educación y Ciencia). We wish to thank Javier Nogales for beneficial discussions.

References

[AL94] William W. Adams and Philippe Loustaunau. *An introduction to Gröbner bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1994.

[Bie96] Daniel Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74(2, Ser. A):121–140, 1996.

[BL07] P. Bonami and M.A. Lejeune. An exact solution approach for portfolio optimization problems under stochastic and integer constraints. IBM Research Report RC24215 (W0703-049), IBM Research Division, Thomas J. Watson Research Center, March 2007.

[BPT00] D. Bertsimas, G. Perakis, and S. Tayur. A new algebraic geometry algorithm for integer programming. *Management Science*, 46(7):999–1008, 2000.

[BW05] D. Bertsimas and R. Weismantel. *Optimization over Integers*. Dynamic Ideas, 2005.

[CF07] M. Corazza and D. Favaretto. On the existence of solutions to the quadratic mixed-integer mean-variance portfolio selection problem. *European Journal of Operational Research*, 176(3):1947–1960, 2007.

[Chn09] Michael T. Chng. Economic linkages across commodity futures: Hedging and trading implications. *Journal of Banking & Finance*, 33:958–970, 2009.

[CLO05] David A. Cox, John Little, and Donal O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer, New York, second edition, 2005.

[CT91] Pasqualina Conti and Carlo Traverso. Buchberger algorithm and integer programming. In *Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991)*, volume 539 of *Lecture Notes in Comput. Sci.*, pages 130–139. Springer, Berlin, 1991.

[Dye92] Martin E. Dyer. A class of convex programs with applications to computational geometry. In *Symposium on Computational Geometry*, pages 9–15, 1992.

[GK87] Monique Guignard and Siwhan Kim. Lagrangean decomposition: a model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 1987.

[GK08] H. Geman and C. Kharoubi. WTI crude oil futures in portfolio diversification: The time-to-maturity effect. *Journal of Banking and Finance*, 32(12):2553–2559, December 2008.

[JHLM01] N. J. Jobst, M. D. Horniman, C. A. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1(5):489–501, 2001.

[KTH79] M. K. Kozlov, S. P. Tarasov, and L. G. Hačijan. Polynomial solvability of convex quadratic programming. *Dokl. Akad. Nauk SSSR*, 248(5):1049–1051, 1979. Translated in Sov. Math., Dokl. 20, 1108-1111 (1979).

[LC98] H.-L. Li and C.-T. Chang. An approximate approach of global optimization for polynomial programming problems. *European Journal of Operational Research*, 107(3):625–632, 1998.

[LT08] Han-Lin Li and Jung-Fa Tsai. A distributed computation algorithm for solving portfolio problems with integer variables. *European Journal of Operational Research*, 186(2):882–891, 2008.

[Mar52] Harry M. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[Mar00] Harry M. Markowitz. *Mean-variance analysis in portfolio choice and capital markets.* Wiley, New York, 2000.

[Mit03] H. D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Math. Program.*, 95(2, Ser. B):407–430, 2003. Computational semidefinite and second order cone programming: the state of the art.

[MM91] Philippe Michelon and Nelson Maculan. Lagrangean decomposition for integer nonlinear programming with linear constraints. *Mathematical Programming*, 52(2, Ser. B):303–313, 1991.

[MM08] Richar O. Michaud and Robert O. Michaud. *Efficient asset management: a practical guide to stock portfolio optimization and asset allocation.* Oxford University Press, New York, 2008.

[Sha71] W. Sharpe. A linear programming approximation for the general portfolio analysis. *Journal of Finance and Quantitative Analysis*, 6:1263–1275, 1971.

[SM05] Bernd Scherer and R. Douglas Martin. *Introduction to modern portfolio optimization with NUOPT and S-PLUS*. Springer, New York, 2005.

[SNT85] Yoshikazu Sawaragi, Hirotaka Nakayama, and Tetsuzo Tanino. *Theory of multiobjective optimization*, volume 176 of *Mathematics in Science and Engineering*. Academic Press Inc., Orlando, FL, 1985.

[Sto73] B. Stone. A linear programming formulation of the general portfolio selection model. *Journal of Financial and Quantitative Analysis*, 8:621–636, 1973.

[Stu96] Bernd Sturmfels. *Gröbner bases and convex polytopes*, volume 8 of *University Lecture Series*. American Mathematical Society, Providence, RI, 1996.

[tt08] 4ti2 team. 4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces. Available at www.4ti2.de, 2008.

[TTN95] Sridhar R. Tayur, Rekha R. Thomas, and N. R. Natraj. An algebraic geometry algorithm for scheduling in presence of setups and correlated demands. *Mathematical Programming*, 69(3, Ser. A):369–401, 1995.

[YK91] H. Yamakozi and H. Konno. Mean absolute deviation portfolio optimization model and its application to Tokyo stock market. *Management Science*, 37:519–531, 1991.

[You98] M.R. Young. A minimax portfolio selection rule with linear programming solution. *Management Science*, 5(44):637–683, 1998.

F. Castro Jiménez. Dpto. de Álgebra, Univ. de Sevilla, Apdo. 1160, 41080 Sevilla, e-mail: `castro@us.es`

J. Gago Vargas. Dpto. de Álgebra, Univ. de Sevilla, Apdo. 1160, 41080 Sevilla, e-mail: `gago@us.es`

I. Hartillo Hermoso. Dpto. de Matemática Aplicada I, Univ. de Sevilla, E.T.S. de Ingeniería Informática, Avda. de Reina Mercedes, s/n, 41012 Sevilla, Spain, e-mail: `hartillo@us.es`

J. Puerto Albandoz, Dpto. de Estadística e Investigación Operativa, Univ. de Sevilla, Facultad de Matemáticas, C/ Tarfia, s/n, 41012 Sevilla, e-mail: `puerto@us.es`

J.M. Ucha Enríquez. Dpto. de Álgebra, Univ. de Sevilla, Apdo. 1160, 41080 Sevilla, e-mail: `ucha@us.es`

Appendix

Procedure NewPolytope(polytope P, matrix Q, point $\boldsymbol{p}_e$, Tol, r_0, $MaxNumCuts$)

```
$NumCuts = 0$ ;
$\boldsymbol{p}_{\max}, r_m^2 =$ ComputeMaxRisk$(P, Q)$ ;
                    /* solve the problem $\max Q(\boldsymbol{x}), \boldsymbol{x} \in P$.  */;
while $r_m^2 - r_0^2 > Tol$ and $NumCuts \leq MaxNumCuts$ do
    $s = \boldsymbol{p}_e + \lambda(\boldsymbol{p}_{\max} - \boldsymbol{p}_e), \lambda \geq 0$ ;
    $\boldsymbol{p}' = s \cap Q$ ;
    $\boldsymbol{v} =$ TangentToQuadric$(Q, \boldsymbol{p}')$ ;
    $DirApprox =$ Round$(\boldsymbol{v}, Prec)$ ;
                    /* round with number of digits = $Prec$ */;
    $Coef =$ TangentToQuadricV$(DirApprox, Q)$ ;
     /* independent term of the tangent hyperplane $Q$ and normal vector equal
    to $DirApprox$ */ ;
    $Coef =$ Ceil$(Coef)$ ;
          /* the best integer to leave the quadric in a half-space */ ;
    $H := DirApprox^t \boldsymbol{x} - Coef$ ;
                              /* new linear cut */ ;
    $NumCuts = NumCuts + 1$;
    $P =$ Polytope$(P, H)$ ;
                    /* add a new cut to polytope $P$  */ ;
    $\boldsymbol{p}_{\max}, r_m^2 =$ ComputeMaxRisk$(P, Q)$ ;
end
return $P$
```

Algorithm 1: DiscreteOptimum

```
Data: budget $B$, risk $r_0^2$, matrix $Q$, vector $\boldsymbol{a}$, vector $\boldsymbol{\mu}$, $MaxNumCuts$,
      $MaxNumNodes, Tol$
Result: $Optimum$, $NumNodesProc$
$\boldsymbol{p}_c = \texttt{ComputeContinuousOptimum}(B, r_0^2, Q, \boldsymbol{a}, \boldsymbol{\mu}), g_c = \boldsymbol{\mu}^t \boldsymbol{p}_c, \alpha = 1/2$ ;
$\boldsymbol{p}_e, g_e = \texttt{ComputeDiscreteApprox}(\boldsymbol{p}_c, B, \boldsymbol{a}, r_0^2, Q)$ ;
$SwEOP = \text{false}, SwFictBounds = \text{false}, SwImprove = \text{false}, SwNumNodes =$
$\text{false}, ListOfVariables = (1 : \dim)$ ;
while not $SwEOP$ do
    if not $SwFictBounds$ then
        $\boldsymbol{b}, ListOfVariables = \texttt{ComputeLowerBounds}(ListOfVariables, \boldsymbol{p}_e)$ ;
    end
    $P = \texttt{Polytope}(\boldsymbol{a}^t \boldsymbol{x} \leq B, \boldsymbol{\mu}^t \boldsymbol{x} \leq g_c, \boldsymbol{\mu}^t \boldsymbol{x} \geq g_d, \boldsymbol{x} \geq \boldsymbol{b})$ ;
    $NumNodesProc = 0$ ;
    while not ( $SwEOP$ or $SwImprove$ or $SwNumNodes$) do
        $P = \texttt{NewPolytope}(P, Q, \boldsymbol{p}_e, Tol, r_0, MaxNumCuts)$ ;
        $G = \texttt{ComputeTestSet}(P), \boldsymbol{p}_{\text{ini}} = \texttt{Reduce}(\boldsymbol{p}_{\text{bounds}}, G)$ ;
        $SwNumNodes, SwImprove, Optimum =$
        $\texttt{TreeSearch}(\boldsymbol{p}_{\text{ini}}, G, Q, \boldsymbol{a}, B, r_0^2, \boldsymbol{b}, \boldsymbol{p}_e, MaxNumNodes, SwFictBounds)$ ;
        if $SwFictBounds$ then
            if not $SwImprove$ then
                $SwEOP = \text{true}$ ;
            else
                $\boldsymbol{p}_e = Optimum, g_e = \boldsymbol{\mu}^t Optimum,\ SwFictBounds = \text{false}$ ;
            end
        else
            if not $SwNumNodes$ then
                $SwEOP = \text{true}$ ;
            else
                if not $SwImprove$ then
                    $\boldsymbol{b} = \boldsymbol{b} + \alpha(\boldsymbol{p}_e - \boldsymbol{b}),\ SwFictBounds = \text{true}$ ;
                end
            end
        end
    end
end
```